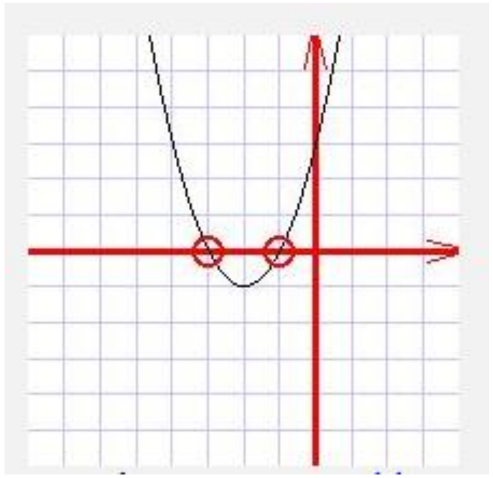
The Quadratic Equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$



Quadratic Explained

The Quadratic Formula Explained

Often, the simplest way to solve "$ax^2 + bx + c = 0$" for the value of x is to factor the quadratic, set each factor equal to zero, and then solve each factor. But sometimes the quadratic is too messy, or it doesn't factor at all, or you just don't feel like factoring. So, while factoring may not always be successful, the Quadratic Formula can *always* find the solution.
The Quadratic Formula uses the "a", "b", and "c" from "$ax^2 + bx + c$", where "a", "b", and "c" are just numbers. The Formula is derived from the process of completing the square, and is formally stated as:

Note that, for the Formula to work, you *must* have "(quadratic) = 0". Note also that the "$2a$" at the bottom of the Formula is underneath *everything* above, not just the square root. And don't forget that it's a "$2a$" under there, not just a "2"! And make sure that you are careful not to drop the square root or the "plus/minus" in the middle of your calculations, or I can guarantee that you will forget to "put them back" on your test, and you'll mess yourself up. And remember that "b^2" means "the square of ALL of b, including the sign", so don't leave b^2 being negative, even if b is negative, because the square of a negative is a positive. In other words, don't be sloppy and don't try to take shortcuts, because it will only hurt you in the long run. Trust me on this!

Breaking Up A large Program Into Smaller Sections

$$x = \frac{-b \pm \sqrt{\boxed{b^2 - 4ac}}}{2a}.$$

This portion of the program needs to be tested to ensure it is not a negative number. Create a separate function for this.

This function contains 3 parameters a, b, and c

```
Function getUnderRoot(ByVal a As Integer, ByVal b As Integer, ByVal c As Integer) As Integer
    Dim ur As Integer
    bSquared = Pow(b, 2)
    ur = bSquared - (4 * a * c)

    If (underRoot < 0) Then
        MsgBox("Cannot find square root of negative number")
    End If
    Return ur
End Function
```

function call

```
underRoot = getUnderRoot(a, b, c)
```

The Numerator

This can now be re-visualized ..

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

like this…

$$x = \frac{-b \pm \sqrt{\text{underRoot}}}{2a}$$

Finding the Root of ***underRoot***

```
Function getRoot(ByVal underRoot As Integer) As Double
    Dim r As Double
    r = Sqrt(underRoot)
    Return r
End Function
```

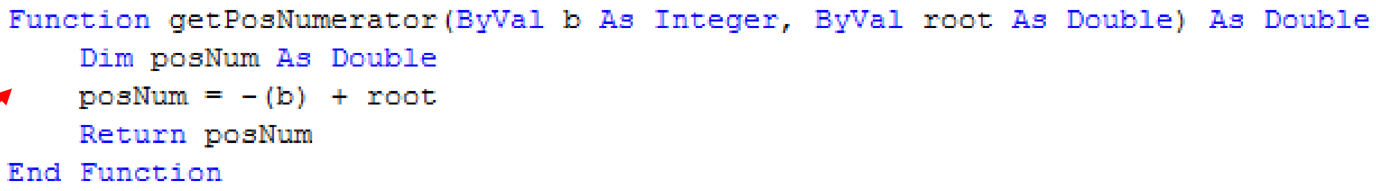
allows us to represent our evolving program as…

$$x = \frac{-b \pm \text{Root}}{2a}$$

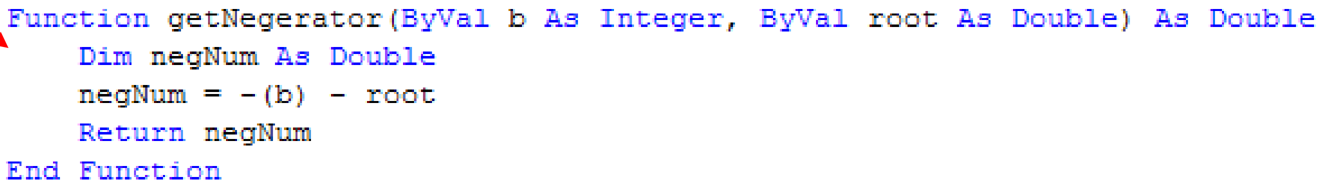
Numerator Plus & Minus

The plus/minus section of the quadratic equation forces us to split the numerator into 2 sections. We will create functions for both and assign the return values to variables that we will call posNumerator and negNumerator.

Functions



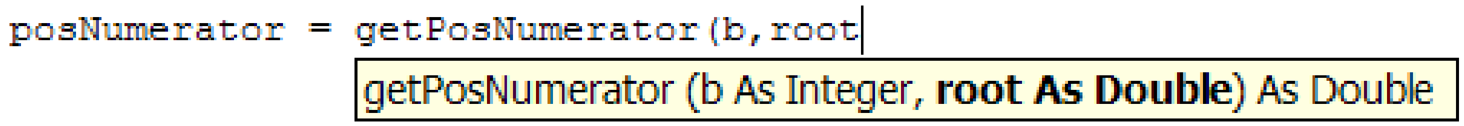
```
Function getPosNumerator(ByVal b As Integer, ByVal root As Double) As Double
    Dim posNum As Double
    posNum = -(b) + root
    Return posNum
End Function
```



```
Function getNegerator(ByVal b As Integer, ByVal root As Double) As Double
    Dim negNum As Double
    negNum = -(b) - root
    Return negNum
End Function
```

Function Calls ***getPosNumerator()*** and ***getNegNumerator()***

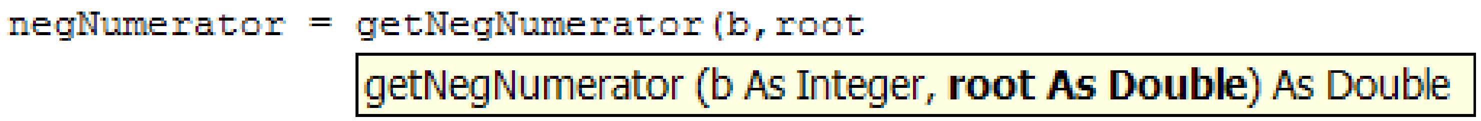
Once you have created a function, a rather interesting feature of VB is, that when you start typing in the function call, the arguments required to match the parameters of that function pop up.



```
posNumerator = getPosNumerator(b,root
```

getPosNumerator (b As Integer, **root As Double**) As Double

Pop up of arguments required to match parameters



```
negNumerator = getNegNumerator(b,root
```

getNegNumerator (b As Integer, **root As Double**) As Double

Plus or Minus

$$x = \frac{\text{posNumerator}}{2\alpha}$$

$$x = \frac{\text{negNumerator}}{2\alpha}$$

Denominator

function call

```
denominator = getDenominator(a)
```

function

```
Function getDenominator(ByVal a As Integer) As Integer
    Dim denom As Integer
    If (a <> 0) Then
        denom = 2 * a
    Else
        MsgBox("Cannot divide by 0")
    End If
    Return denom
End Function
```

Almost Finished

$$x = \frac{posNumerator}{denominator}$$

$$x = \frac{negNumerator}{denominator}$$

Now create a double variable to hold each of the possible values.

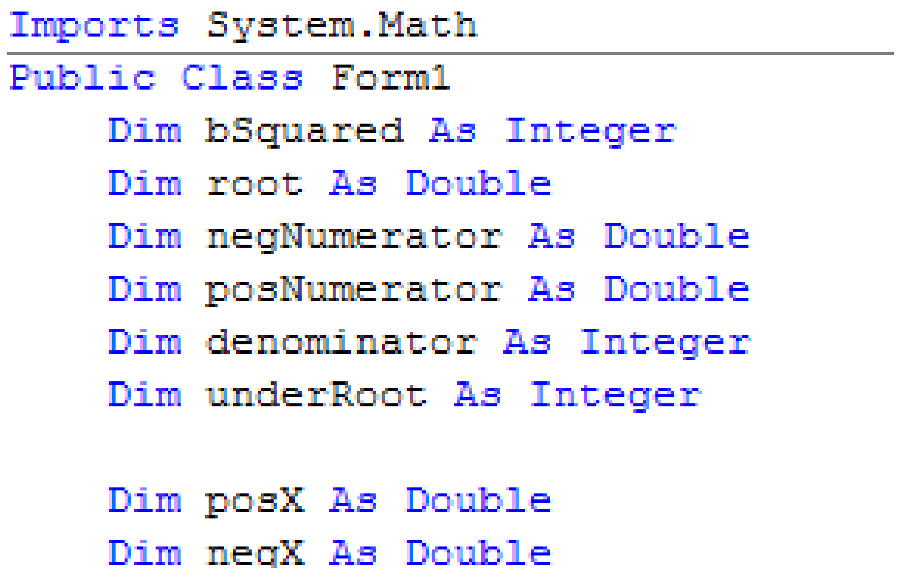
```
Dim posX As Double
Dim negX As Double
```

$$posX = \frac{posNumerator}{denominator}$$

$$negX = \frac{negNumerator}{denominator}$$

The Finished Program

The Form Declarations



```
Imports System.Math
Public Class Form1
    Dim bSquared As Integer
    Dim root As Double
    Dim negNumerator As Double
    Dim posNumerator As Double
    Dim denominator As Integer
    Dim underRoot As Integer

    Dim posX As Double
    Dim negX As Double
```

The Functions

```
Function getUnderRoot(ByVal a As Integer, ByVal b As Integer, ByVal c As Integer) As Integer
    Dim ur As Integer
    bSquared = Pow(b, 2)
    ur = bSquared - (4 * a * c)

    If (underRoot < 0) Then
        MsgBox("Cannot find square root of negative number")
    End If
    Return ur
End Function
```

```
Function getRoot(ByVal underRoot As Integer) As Double
    Dim r As Double
    r = Sqrt(underRoot)
    Return r
End Function
```

```
Function getPosNumerator(ByVal b As Integer, ByVal root As Double) As Double
    Dim posNum As Double
    posNum = -(b) + root
    Return posNum
End Function
```

```
Function getNegNumerator(ByVal b As Integer, ByVal root As Double) As Double
    Dim negNum As Double
    negNum = -(b) - root
    Return negNum
End Function
```

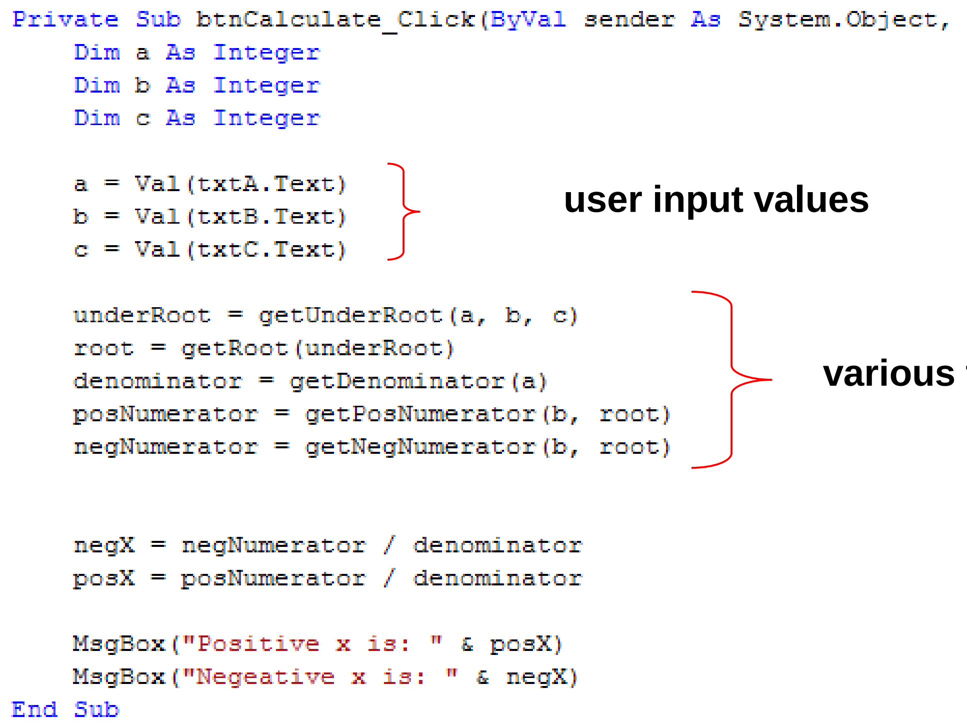
```
Function getDenominator(ByVal a As Integer) As Integer
    Dim denom As Integer
    If (a <> 0) Then
        denom = 2 * a
    Else
        MsgBox("Cannot divide by 0")
    End If
    Return denom
End Function
```

The Calculate Button

btnCalculate is the starting point for the program. From here the initial values are gathered.

What makes this Sub() interesting is what it doesn't do.
It does very little in the way of actual calculations. Instead it is the point from which all the functions are called.

Like a good executive that efficiently delegates tasks to many different employees it oversees the larger picture of the program and does not deal with the details.

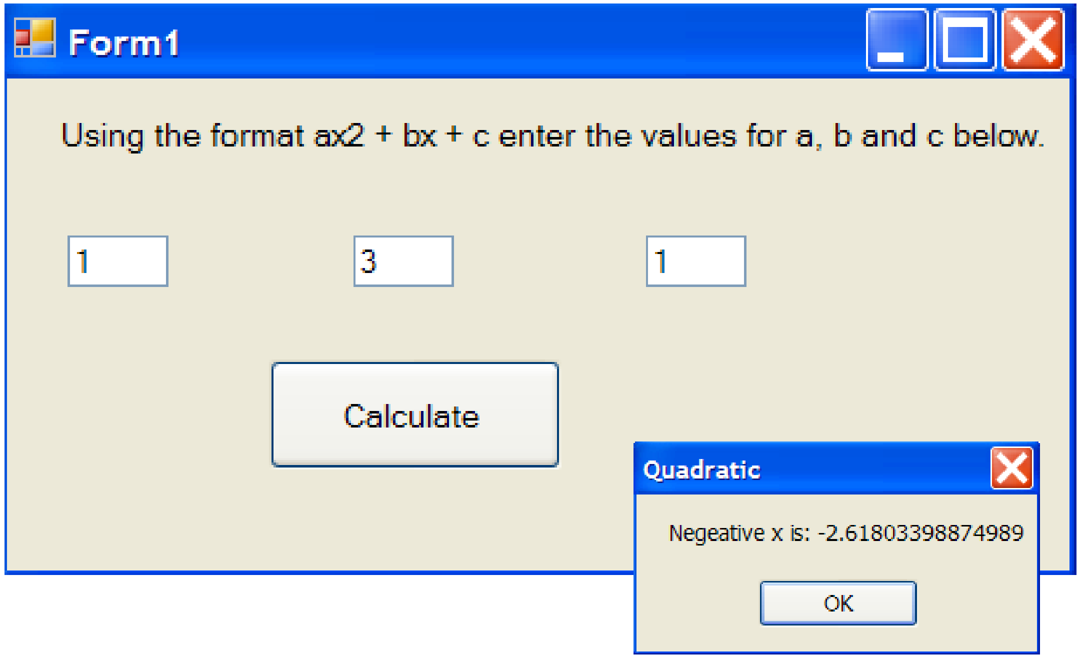


```
Private Sub btnCalculate_Click(ByVal sender As System.Object,
    Dim a As Integer
    Dim b As Integer
    Dim c As Integer

    a = Val(txtA.Text)
    b = Val(txtB.Text)
    c = Val(txtC.Text)

    underRoot = getUnderRoot(a, b, c)
    root = getRoot(underRoot)
    denominator = getDenominator(a)
    posNumerator = getPosNumerator(b, root)
    negNumerator = getNegNumerator(b, root)


    negX = negNumerator / denominator
    posX = posNumerator / denominator

    MsgBox("Positive x is: " & posX)
    MsgBox("Negeative x is: " & negX)
End Sub
```

user input values

various function calls

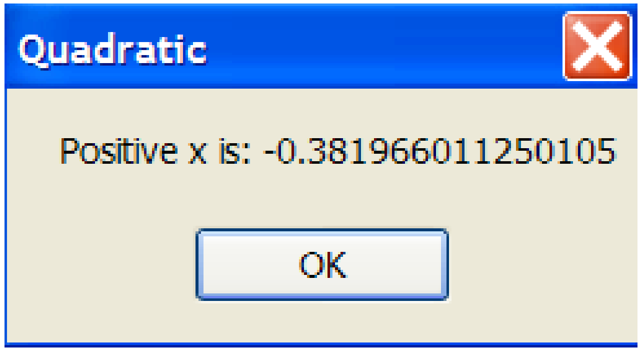
The Program



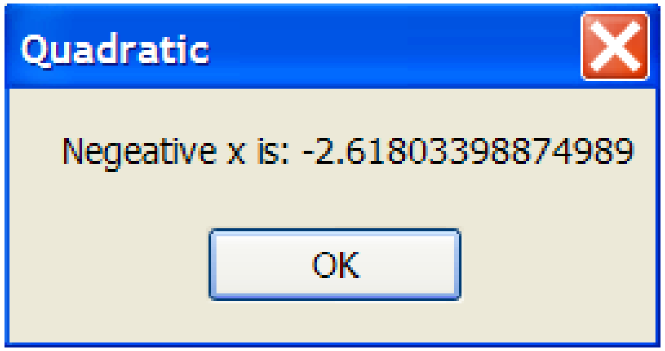
Form1

Using the format ax2 + bx + c enter the values for a, b and c below.

```
Negeative x is: -2.61803398874989
```



```
Positive x is: -0.381966011250105
```



```
Negeative x is: -2.61803398874989
```